15. Timers en klokken

Timers

Timers hebben veel toepassingen. We kunnen ze gebruiken om een tijdsinterval te meten, om PWM signalen te genereren of om een signaal te geven als een tijdsduur verstreken is. Voorlopig heeft MicroPython alleen de eenvoudige toepassing geïmplementeerd: het periodiek oproepen van een functie Voor het werken met een timer gebruiken we **klasse Timer** uit library machine.

Declaratie van de timer

De RP2040 heeft één hardware timer, MicroPython gebruikt die voor interne processen. In een programma kunnen we een software timer gebruiken. We declareren een vertegenwoordiger met

```
tim = Timer()
```

De esp32 en de esp-c3 hebben vier hardware timers: 0, 1, 2 en 3. We declareren een vertegenwoordiger van timer n als

```
Tim = Timer(n)
```

Voor de ESP8266 is één timer beschikbaar. Je declareert hem met

```
Tim = Timer(-1)
```

De SAMD21 versie van MicrPython ondersteunt de timer (nog) niet.

Verder werken met Timers

We initialiseren de timer met methode init:

```
tim.init(mode = Timer.PERIODIC)
tim.init(mode = Timer.ONE_SHOT)
tim.init(freq = 1)
tim.init(period=1000)
tim.init(callback = tik)
```

De **mode** bepaalt de toepassing:

- Mode ONE_SHOT: na één periode geeft de timer een signaal, een interrupt de het lopende programma onderbreekt om een functie uit te voeren.
- Mode PERIODIC: de timer blijft lopen, na elke periode volgt een interrupt en begint alles opnieuw.

Callback bepaalt de functie die wordt uitgevoerd na een timerperiode

freq of **period** bepalen na welke tijd de timerfunctie wordt uitgevoerd:

- period = 1000: na elke 1000 ms
- freq =1: één maal per seconde

In het eerste voorbeeld gebruiken we een timer om een LED te laten flikkeren, één maal per seconde.

```
#------------------------------#
#  timer-led.py                #
#                              #
#  12 december 2021            #
#  Dirk Ghysels                #
#------------------------------#
from machine import Pin, Timer
led = Pin(25, Pin.OUT)
ledAan = 1

def tik(timer):
    global led, ledAan
    ledAan = 1-ledAan
    led.value(ledAan)

#tim = Timer(1)  #voor esp32 en esp32-c3
tim = Timer(-1)  #voor esp8266
#tim = Timer()  #voor rp2040

tim.init(freq = 1, mode = Timer.PERIODIC, callback = tik)
```

Elke seconde zal de timer functie tik oproepen, die verandert de status van de LED. Een volledige aan/uit cyclus duurt dus 2 seconden.

Het resultaat voor de esp32-c3 is zeer onnauwkeurig. Op de website van MicroPython is hierover geen informatie te vinden.

- led is een globale variabele, die verwijst naar GPIO 25 als digitale uitgang. Globale variabele ledAan geeft aan of de LED al dan niet aan is.
- in het hoofdprogramma definiëren we tim, een periodieke timer met frekwentie 1 Hz. De callbackfunctie is tik. De timer roept die één maal per seconde aan.
- Functie tim verandert ledAan (0 wordt 1, 1 wordt 0) en verandert daarmee de toestand van de LED.

Project *KLOK*

De tijd berekenen

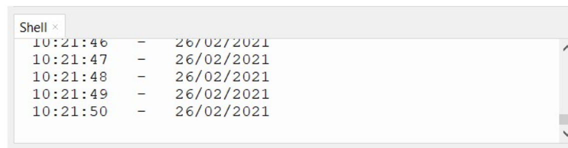
Met een secondetimer kunnen we een klok programmeren. Functie klik verhoogt bij elke tik de secondenteller. Dat doen we tot 60, dan wordt de secondenteller nul en verhogen we de minutenteller. Daarna de urenteller, eventueel dagenteller, maandteller en jaarteller. We gaan uit van beginwaarden voor de tellers. Dat is een primitieve manier om de tijd in te stellen. Met enkele toetsen en een uitbreiding van het programma kan je dat verbeteren.

```
#------------------------------#
#  timer-klok.py               #
#                              #
#  23 februari 2021            #
#  Dirk Ghysels                #
#------------------------------#
from machine import Pin, Timer

sec = 55
min = 59
uur = 23
dag = 30
maand = 4
jaar = 2020

def tik(timer):
    global sec, min, uur, dag, maand, jaar
    dagaantal = 31
    if (maand in (4, 6, 911)):
        dagaantal = 30
    if ((maand==2)and (jaar%4==0)):
        dagaantal = 29
    if ((maand==2)and (jaar%4!=0)):
        dagaantal = 28
    sec += 1
    if (sec==60):
        sec = 0
        min += 1
        if (min==60):
            min = 0
            uur += 1
            if (uur==24):
                uur=0
                dag += 1
                if (dag == dagaantal+1):
                    dag = 1
                    maand += 1
                    if (maand == 13):
                        maand = 1
                        jaar += 1
    print ("%02u:%02u:%02u - %02u/%02u/%u" %(uur,min,sec,dag,maand,jaar))

tim = Timer(1)  #voor esp32
#tim = Timer()  #voor rp2040
tim.init(freq = 1, mode = Timer.PERIODIC, callback = tik)
```



Figuur 87: Klok

Het hoofdprogramma bestaat uit de laatste twee regels. Met klasse Timer maken we object tim, die initialiseren we in de laatste regel. Alle berekeningen gebeuren in interrupt routine tik. We tellen seconden tot 60, …. Het aantal dagen van een maand is variabel, het programma houdt rekening met schrikkeljaren. Februari heeft 29 dagen als het jaartal deelbaar is door 4. Een uitzondering is 2100, dat is geen schrikkeljaar maar daar houden we geen rekening mee.

Met dit voorbeeld kan je met een MicroPython en een display mooie klokken bouwen. Er bestaan veel displays voor microcontrollersystemen. In dit project gebruiken we het 4-digit 7-segment display met seriële aansturing uit vorig hoofdstuk.

Een klok met 7-segment display

We maken de klok door vorige programma te combineren met de aansturing van het display uit vorig hoofdstuk. Het display toont alleen minuten en uren, datum kunnen we weglaten.

```
#------------------------------#
#  tm1637-klok.py              #
#                              #
#  29 maart 2021               #
#  Dirk Ghysels                #
#------------------------------#
from machine import Pin, Timer
import tm1637, utime
buttonMin = Pin(14, Pin.IN)
buttonUur = Pin(15, Pin.IN)

sec = 55
min = 45
uur = 10

def min_handler (pin):
    global min
    min += 1
    if min>59:
        min = 0
    mydisplay.numbers(uur, min)
    utime.sleep (1)
def uur_handler (pin):
    global uur
    uur += 1
    if uur>23:
        uur = 0
    mydisplay.numbers(uur, min)
```

```
    utime.sleep(1)

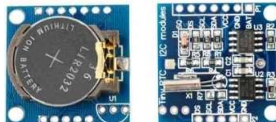
def tik(timer):
    global sec, min, uur
    sec += 1
    if (sec==60):
        sec = 0
        min += 1
        if (min==60):
            min = 0
            uur += 1
            if (uur==24):
                uur=0
        print ("%02u:%02u" %(uur,min))
        mydisplay.numbers(uur, min)

mydisplay = tm1637.TM1637(clk=Pin(16), dio=Pin(17))
tim = Timer(1)  #voor esp32
#tim = Timer()  #voor rp2040
tim.init(freq = 1, mode = Timer.PERIODIC, callback = tik)
buttonMin.irq (trigger=Pin.IRQ_FALLING, handler=min_handler)
buttonUur.irq (trigger=Pin.IRQ_FALLING, handler=uur_handler)
```

We hebben twee drukknoppen toegevoegd. ButtonMin, tussen GPIO14 en Vcc stelt de minuten in, de tweede, tussen GPIO15 en Vcc de uren. Ze genereren IRQ's met handlers resp. min_handler en uur_handler. Om contactdender te vermijden, gebruiken we de schakeling van hoofdstuk *Hardware Interfacing*. De weerstand in die schakeling is de pull-up weerstand, de poort krijgt waarde 0 bij indrukken, de interrupt wordt getriggerd bij het nul-worden op de pin.

De RTC van de controller

Een **RTC** of **realtimeklok** is een computerklok, meestal in de vorm van een IC die de tijd bijhoudt. Een RTC heeft meestal een batterij zodat hij blijft verder werken, ook als de computer uit staat. Je vindt je deze klokken in alle soorten computers, ook in veel embedded systemen. Een veel gebruikt RTC-IC is de DS1307 van Maxim, die wordt ook verkocht als een module met batterij.



Figuur 88: RTC, DS1307

De meeste 32-bit controllers hebben een ingebouwde RTC. Die werkt onafhankelijk van de rest van de controller. De RTC berekent jaar (0 … 4095), maand (1 … 12), dag (1 tot 28, 29, 30 of 31, afhankelijk van de maand), uur (0 … 23), minuut (0 … 60 en seconde (0 … 60). Met sofware kunnen we de klok uitlezen en instellen.

De meeste computersystemen behandelen tijd als het aantal seconden sinds 1 januari 1970, 0:00, UTC. (UTC=Greenwich tijd). De controller houdt die tijd bij in de RTC. Met klassen utime en RTC uit library machine zal MicroPython de RTC instellen en uitlezen. Voor de omrekening van tijdnotaties zijn er functies: omzetten van aantal seconden sinds 1/1/1970 naar jaar, maand, …, minuten en seconden.

Klasse RTC

Deze klasse bestuurt de RTC.

- **rtc = RTC()** creëert een RTC-object.
- **rtc.datetime()** geeft de huidige tijd als een tuple
- Bij het opstarten van een programma met Thonny, geeft Thonny de juiste waarden door aan de RTC.
- Je kunt de RTC instellen met
  rtc.datetime((2017, 8, 23, 1, 12, 48, 0, 0))
  Dit is 23/8/2017, maandag, 12/48/0, 0 µs. *Let op: dubbele haken!*
- Library RTC is niet geïmplementeerd voor de SAMD21

Het programma hieronder toont de tijd, éénmaal per seconde.

Het programma hieronder werkt zonder aanpassingen op en RP2040, een ESP8266, een ESP32 en een ESP32-C3.

```
#------------------------------#
#  rtc-rp2040.py               #
#                              #
#  21 juni 2021                #
#  Dirk Ghysels                #
#------------------------------#
from machine import RTC
rtc = machine.RTC()
sec0 = 0
while True:
    nu = rtc.datetime()
    jaar = nu[0]
    maand = nu[1]
    dag = nu[2]
    uur = nu[4]
    minuut = nu[5]
    seconde = nu[6]
    if (seconde != sec0):
        print("%02u:%02u:%02u  %02u/%02u/%u" %(uur, minuut, seconde, dag, 
maand, jaar))
        sec0 = seconde
```

```
Shell
MicroPython v1.16 on 2021-06-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT
 10:10:12  21/06/2021
 10:10:13  21/06/2021
 10:10:14  21/06/2021
 10:10:15  21/06/2021
 10:10:16  21/06/2021
 10:10:17  21/06/2021
 10:10:18  21/06/2021
 10:10:19  21/06/2021
 10:10:20  21/06/2021
 10:10:21  21/06/2021
 10:10:22  21/06/2021
 10:10:23  21/06/2021
```

Figuur 89: uitvoer van rtc-rp2040.py

Klasse utime

Met utime kan je de tijd opvragen en omzetten naar meer gebruikelijke vormen. Enkel functies van utime zijn:

- **utime.time**() geeft de tijd in seconden. Als je dat getal deelt door het aantal seconden per minuut, het aantal minuten per uur, het aantal uren per dag en het aantal dagen per jaar dan zie je het aantal jaren sinds 1 januari 1970:

```
>>> y = utime.time()
>>> print (y)
 1616666282
>>> jaar = y/60/60/24/365
>>> print(jaar)
 51.26415
```

Figuur 90: utime.time()

- **utime.localtime**() geeft de huidige tijd weer. In het voorbeeld hieronder zie je jaar, maand, dag, uur, minuut en seconde. Het laatste getal, 66 in het voorbeeld hieronder, zegt dat het nu in de 66ste dag van het jaar is.

```
>>> time = utime.localtime()
>>> print(time)
 (2021, 3, 7, 11, 22, 38, 6, 66)
>>> jaar = time[0]
>>> print (jaar)
 2021
```

Figuur 91: utime.localtime()

- **utime.sleep**(n) laat het systeem n seconden wachten.
- **utime.sleep_ms**(n) laat het systeem n milliseconden wachten.
- **utime.sleep_us**(n) laat het systeem n microseconden wachten.
- **utime.ticks_ms**() geeft de tijd inmilliseconden t.o.v. een onbekend beginpunt. Dat getal op zich betekent niets, maar met een verschil van twee ticks kunnen we tijdsintervallen meten.
- **utime.ticks_us**() is hetzelfde, maar in microseconden.

- **utime.ticks_diff**(ticks1, ticks2) berekent de tijd tussen ticks1 en ticks2.

Een klok, gebaseerd op de RTC en een 7-segment volume vind je hieronder:

```
#------------------------------#
#  RTC-tm1637.py               #
#                              #
#  21 juni 2021                #
#  Dirk Ghysels                #
#------------------------------#
from machine import Pin, Timer, RTC
import tm1637, utime
buttonMin = Pin(14, Pin.IN)
buttonUur = Pin(15, Pin.IN)
min = 33
uur = 12
sec = 0

rtc = machine.RTC()
time = rtc.datetime

def min_handler (pin):
    time = rtc.datetime()
    uur = time[4]
    min = time[5]
    min += 1
    if min>59:
        min = 0
    rtc.datetime((2021,2,14,4,uur, min, sec, 0))
    mydisplay.numbers(uur, min)

def uur_handler (pin):
    time = rtc.datetime()
    uur = time[4]
    min = time[5]
    uur += 1
    if uur>23:
        uur = 0
    rtc.datetime((2021,2,14,4,uur, min, sec, 0))
    mydisplay.numbers(uur, min)

buttonMin.irq (trigger=Pin.IRQ_FALLING, handler=min_handler)
buttonUur.irq (trigger=Pin.IRQ_FALLING, handler=uur_handler)
mydisplay = tm1637.TM1637(clk=Pin(16), dio=Pin(17))
while(True):
    time = rtc.datetime()
    uur = time[4]
    min = time[5]
    mydisplay.numbers(uur, min)
    utime.sleep(5)
```

Hier gebruiken we knoppen om de klok in te stellen. Contactdender in deze toepassing zou heel vervelend zijn. Een druk op de minutenknop geeft meerdere pulsen, het aantal minuten worden verhoogd met 1, 2, 3 of meer. Het gelijkzetten is hierdoor bijna onmogelijk. Gebruik de schakeling uit hoofdstuk *Hardware interfacing* om contactdender te vermijden.

Opmerking:

Na een stroomuitval en herstart geeft deze klok verkeerde waarden. Met een noodvoeding blijft de klok doorwerken.